Attorney Docket No. 2860-065; P2287/TJC

APPLICATION FOR UNITED STATES LETTERS PATENT FOR

TECHNIQUES FOR IMPROVING VIRTUAL CHANNEL MANAGEMENT AND MAINTENANCE IN A NETWORK ENVIRONMENT

INVENTORS:
Amit Gupta
Raphael Rom

PREPARED BY:
David L. Stewart
LOWE PRICE LEBLANC & BECKER
99 Canal Center Plaza, Suite 300
Alexandria, VA 22314
Telephone: (703) 684-1111
Facsimile: (703) 684-1124

2860-065

P2287/TJC

TECHNIQUES FOR IMPROVING VIRTUAL CHANNEL MANAGEMENT AND MAINTENANCE IN A NETWORK ENVIRONMENT

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates to communication systems and, more particularly to the creation, maintenance, use and deletion of virtual channels in a network environment.

Description of Related Art

Modern data networks trace their historical origins to telephone switching networks. Telephone switching networks set up connections between an originating caller and a called subscriber selected by dialing the called subscriber's telephone number.

Digital networks are known which permit information packets to be routed from a source to a destination without the necessity for a hardwired connection between the calling subscriber and the destination subscriber. Packets are routed over links between nodes of a network. Rather than a physical connection, a virtual connection,

sometimes called a virtual channel or virtual circuit is defined between the source of the packet and its destination. A virtual circuit gives the appearance of maintaining a hardwire connection, but utilizes the resources of the connection only when data need to be sent. This permits the link to be shared by other virtual circuits and improves the efficiency and throughput of the links of the network.

One form of modern high speed network uses a set of standards collectively termed asynchronous transfer mode (ATM). The basic data element used by these networks is a cell, which is a sequence of 53 bytes of which the first 5 are the header and the other 48 are the cell payload. To allow very high data rates (in the hundreds of megabytes per second and above) the switching time of a cell in every switch or node along the route through the network must be minimal. ATM networks achieve this by way of a virtual channel mechanism which requires a set-up phase before the actual data stream can be transported from source to destination. The notion of a virtual path is defined in the ATM standard. Each virtual path "contains" 2^{16} (65,536) virtual circuits. Supporting virtual path requires significant additions to the ATM switch hardware. Consequently, many switches only support virtual circuits; they do not support virtual paths.

The Problems

It would be desirable to have an efficient mechanism for quickly allocating multiple connections without requiring extra hardware support in the network nodes. Further, it would be desirable to reduce the table size required for network switches to support large numbers of virtual circuits. It would further be desirable to implement multiple virtual channels set-up while maintaining compatibility with current switches but nevertheless supporting heterogeneity in switches in a network. It would further be desirable to be able to route multiple virtual circuits over different paths. It would also be desirable to support dynamic aggregation of individual virtual circuits into larger entities. It would also be desired to permit enhanced resource sharing across connections of a network. It would further be desirable to have techniques for handling ATM cell interleaving problems in a multi-sender multi-cast. It would also be desirable to have efficient support for fast connection establishment. It would further be desirable to improve the efficiency with which "keep-alive" and "refresh" packets are sent through the network.

SUMMARY OF THE INVENTION

The problems of the prior art are solved in accordance with the invention by providing a new entity for use in networks called the Virtual Circuit Bunch (VCB). Virtual Circuit Bunches can be utilized to set-up and manage together groups of virtual circuits in a flexible way which results in increased performance as well as overcoming the problems of the prior art.

The Virtual Circuit Bunch, as set forth herein, enables groups of Virtual Circuits to be established between one or more points of origin and one or more destinations. Virtual Circuit Bunches can be implemented without any changes to switch hardware and permit great flexibility in Virtual Circuit allocation.

One aspect of the invention is directed to a switching node which includes a switching matrix, and a controller to control the switching matrix, in which the controller is configured to set up group(s) of virtual circuits to one or more destinations as a virtual circuit bunch.

Another aspect of the invention is directed to a method of allocating virtual circuits, by establishing a plurality of virtual circuits from one node to at least one other node as a virtual circuit bunch in response to a single request.

Another aspect of the invention relates to a method of identifying virtual circuits at a node going to a common destination node; and aggregating those virtual circuits into a virtual circuit bunch.

Another aspect of the invention relates to a method of providing a fast connect service in a digital switching network by appropriately assigning a packet to a virtual circuit of a virtual circuit bunch without establishing a connection.

Another aspect of the invention relates to a method of allocating virtual circuits in a switching system allocating a virtual circuit to all nodes participating in a multicast using a virtual circuit bunch.

Another aspect of the invention relates to a system for the transmission of digital communications, including a plurality of user communication devices; a plurality of at least partially interconnected switching nodes, each node serviced by a node controller servicing said user communications devices; in which at least one of said node controllers is configured to set up a group of virtual circuits to respective one or more destinations as a virtual circuit bunch.

Other aspects of the invention are directed to computer program products for carrying out the invention.

The foregoing and other features, aspects and advantages of the present invention will become more

apparent from the following detailed description of the present invention when taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features and advantages of the present invention will be apparent from the following description, in which:

Figure 1 has a block diagram showing an exemplary network used for carrying out the invention.

Figure 2 is a block diagram of an exemplary node of the network shown in **Figure 1**.

Figure 3A is a front view of a computer suitable for use as a control device or controller for one or more network nodes.

Figure 3B is a block diagram showing an exemplary hardware architecture of the controller of **Figure 3A**.

Figure 3C is a view of an exemplary memory medium suitable for storing programs and data used in carrying out the invention.

Figure 4 is a flow chart of a process used at a node for setting up a virtual circuit from an originating station U1 to a destination station U2.

Figure 5 is a time line diagram showing set-up of a virtual circuit between U1 and U2 over the network of **Figure 1**.

Figure 6 is a block diagram showing operation of a network node when implemented as a ATM switch.

Figures 7A, 7B and 7C show exemplary packet formats used in requesting set-up of a Virtual Circuit Bunch (VCB).

Figure 8A is a table summarizing the VCB request of **Figure 7C** and **Figure 8B** shows the same table but with the incoming ports utilized in the VCB assigned to carry traffic.

Figure 9 is a block diagram showing the switches and links used for setting up the VCB requested in **Figure 7C**.

Figure 10 is an exemplary table storing VCB information.

Figure 11 shows exemplary tables storing connection information relating to a VCB used at a node at either end of a VCB.

Figure 12 shows exemplary tables storing connection information relating to a VCB at a node intermediate to the end nodes of a VCB.

Figure 13 is a flow chart of a process for setting up a VCB in accordance with the invention.

Figure 14 is a flow chart of a process for setting up a VC over an established VCB in accordance with the invention.

Figure 15 is a flow chart of a process for breaking down a VCB in accordance with the invention.

Figure 16 is a flow chart of a process for aggregating virtual circuits into a VCB in accordance with the invention.

Figure 17 is a block diagram illustrating statistically sharing virtual circuits of a VCB for a fast connect service.

Figure 18 is a flow chart of a process used for providing a fast connect service.

Figure 19 is a block diagram of a portion of a network used to illustrate split routing of a VCB.

Figure 20 is a block diagram showing use of a VCB when multi-casting over a network.

Figure 21A is a block diagram showing use of a VCB when multi-casting using plural sources over a network.

Figure 21B is a block diagram showing the cell interleaving problem and how it is solved when using a VCB.

NOTATIONS AND NOMENCLATURE

The detailed descriptions which follow may be presented in terms of program procedures executed on a computer or network of computers. These procedural descriptions and representations are the means used by those skilled in the art to most effectively convey the substance of their work to others skilled in the art.

A procedure is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. These steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be noted, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Further, the manipulations performed are often referred to in terms, such as adding or comparing, which are commonly associated with mental operations performed by a human operator. No such capability of a human operator is necessary, or desirable in most cases, in any of the operations described herein which form part of the present invention; the operations are machine operations. Useful machines for performing the operation of the present invention include general purpose digital computers or similar devices.

The present invention also relates to apparatus for performing these operations. This apparatus may be

specially constructed for the required purpose or it may comprise a general purpose computer as selectively activated or reconfigured by a computer program stored in the computer. The procedures presented herein are not inherently related to a particular computer or other apparatus. Various general purpose machines may be used with programs written in accordance with the teachings herein, or it may prove more convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these machines will appear from the description given.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figure 1 is a block diagram showing an exemplary network useful in carrying out the invention. The network shown is comprised of nodes A through J, each node being capable of routing digital data from users connected to the node or from other nodes connected to the node through to a destination. For example, when user U1, which is connected to node A, desires to connect to user U2, connected to node J, the user will request a connection from node A to node J as discussed more hereinafter. In one implementation, node A will do a calculation of a proposed route through the network from user U1 to user U2. In the example shown, node A proposes a route from user 1 through node A, to node C,

to node F, to node H, to node J, to user U2. Different types of networks have different types of routing mechanisms. In the example shown in **Figure 1**, the invention will be described with reference to an ATM network where each of the nodes constitutes an ATM switch. The principles of the invention, however, are applicable to general networks without restriction to the type of switching technology.

Figure 2 is a block diagram of an exemplary node of the network shown in **Figure 1**. Digital data arrives on incoming ports 1 through N over digital links connected to users and other nodes. The incoming digital data is optionally buffered in buffers **240** prior to being routed to the appropriate outgoing port of the set of outgoing ports 1-M. The outgoing ports connect to either end-users or other nodes as desired to implement a particular network topology. One or more interconnect switches ~~**230**~~ forming part of a switching fabric **230** are activated to connect the appropriate incoming port to the appropriate outgoing port so that data is routed from the source toward the destination in an orderly fashion. Data having passed through the switching fabric **230** is optionally applied to outgoing buffers **250** prior to being applied to respective links connected to the outgoing ports. ~~A control device or controller **200.**~~

At a high level, ATM networks can be viewed as a collection of ATM-switches interconnected by high-speed links to provide ATM service to a number of external users. A typical switch is comprised of three main parts: the switching fabric, 230 a buffering unit, 240, 250 and a control unit. The control unit is divided into two parts. The first part 220 controls the cell switching properties, i.e., controls and coordinates the operation of the buffering unit and the switching fabric; for performance reasons it is typically implemented in hardware such as lookup tables, finite state machines, etc. The second part 210 is responsible for all other control functions such as preparing and maintaining the switching tables, coordinating activities with other switches and with network managements facilities, and servicing the individual user requests. This is referred to as the control point (CP) of the switch. As used herein the switch contains the fabric, the buffering units, and the basic core of the control unit, whereas the switch and the CP together will be referred to as a node. The CP is essentially a logical entity which is typically implemented in software although firmware and other software/hardware configuration are also possible.

The device on which the CP is implemented is teamed the Controlling Device (CD) or controller. 205 The CD may not be a single entity, that is, parts of the CP can be

implemented in different hardware pieces. For example, it would be quite common for a CP to be split into two portions the first of which is implemented within the switch and the second in another closely attached device. Another typical split of the CP functions into multiple physical CDs is in cases were each link adaptor contains a separate micro-controller and another optional central processor is responsible for the common switch functions. Another implementation option is a single CD that hosts several CPs, each controlling a different switch and possibly communicate with each other within the same CD. A configuration in which a single CP controls several switches is also possible. For example, if the particular way in which a switch is implemented lends itself to the efficient interconnection of several switches (typically of the same kind) then it would be natural to have this set of switches controlled by a single CP.

Two CPs are called neighbors if they control two switches that are directly connected by a link.

A CP is a uniquely identifiable entity within the network. That is, it is possible to name a CP as the endpoint of a VC and switches can identify cells destined to the CP and forward them appropriately (this is the mechanism by which control messages can be exchanged between CPs.

Every switch in the network possesses a unique ID, which is assigned to it at configuration time. This ID, termed switch ID, is guaranteed to be unique only within the network. Links can have IDs that are local to the switch from which they emanate. This allows to keep the local link IDs (which are frequently used) short. Therefore, to globally identify a link some combination of a CP ID, node ID and link ID must be used. Link, switch and CP IDs appear in the messages that are exchanged between CPs and hence it is desirable to choose as short a descriptor as possible.

In general, CPs perform those functions that are necessary to make a group of switches operate as an integrated network. The process of integration entails understanding of and compensation for the differences between switches of different makes and models. One of the most important functions is that of virtual circuit (connection) management which includes VC setup, maintenance and VC take down.

To perform their functions, CPs obviously need to communicate with one another either directly or indirectly. The control mechanism typically requires CPs to mostly communicate with neighbor CPs. In many cases a message arriving at a CP may trigger the transmission of another message. For example, if a topological change has occurred, a CP will typically notify other CPs of the

change; each of the recipients of this message may, in turn, generate new messages to further distribute the information. A control link as the channel over which two CPs communicate. The collection of control of control links must enable every CP to distribute the appropriate data to every other CP. In a typical design, there will be a control link between any two neighboring CPs.

Figure 3A is a front view of a computer suitable for use as a control device or controller for one or more network nodes. Although the computer is illustrated as a standalone device, computers used as controllers are more typically printed circuit cards which are rack mounted and plug into a backplane. The controller **300** illustrated includes drives **310A** and **310B** for receiving, reading and writing information stored on removable storage media. The controller **300** also may optionally include a CD ROM device, not shown and an internal hard drive, also not shown. Display **320**, keyboard **330** and mouse **340** may be optional in a particular environment although typically, these input output devices may be attached for troubleshooting a particular controller. The actual configuration of a controller may vary from switch to switch.

Figure 3B is a block diagram showing an exemplary hardware architecture of the controller of **Figure 3A**.

As shown in **Figure 3B**, the controller has a CPU **355** which connects and controls bus **350**. A disk controller **370** interfaces the floppy drives **373**, internal hard drive **372** and CD ROM **371**. The particular configuration for storage media may vary from controller to controller. Read Only Memory **360** and Random Access Memory **365** provide memory for storing programs and data information to carry out the controller functions. Interfaces **345** and **375** are available to permit input output devices such as keyboard **330**, mouse **340** and display **320** to be connected when working with the controller for installation, setup, maintenance or the like. A communications port **385** permits external communications from the bus. Switch interface ~~190~~ 390 provides the control and exchange of information between the switch controlled by the controller and the CPU **355** providing the control.

Figure 3C is a view of an exemplary memory medium suitable for storing programs and data use in carrying out the invention.

Figure 4 is a flow chart of a process used at a node in setting up a virtual circuit from an originating station U1 to a destination station U2. This process describes a simple single VC connection between U1 and U2. When a connection request is received on an incoming port i, such as from a user or a predecessor node, it is routed to the controller as discussed previously **(400)**.

The controller consults a network map and determines the (best) next node for routing data from U1 toward its destination address **(410)**. In some networks, the controller will only determine the best next node whereas in others, it will attempt to determine the entire route to the destination. The controller selects an outgoing port and VCI to the next node with adequate resources to service a level of service (LOS) specified for the connection **(420)**. The controller sends a request control packet on the outgoing port and virtual circuit to the next node controller and waits for an acknowledgment **(430)**. The controller may send this over a control link between the two switches or over the actual port and virtual circuit proposed for use in the connection. When an acknowledgment is received from the next node, the acknowledgment will be sent to predecessor node or to the user indicating acceptance of the connection and a willingness to provide the services at the requested LOS specified in the request. The node controller will make entries in the switch table specifying the port and VC to be utilized for the connection **(440)**.

Figure 5 is a time line diagram showing set-up of a virtual circuit between U1 and U2 over the network of Figure 1. User U1 initiates a connection request message to node A **(500)** which requests a virtual circuit to node C **(505)** which requests a virtual circuit to node F **(510)**

which requests a VC to node H **(515)** which requests a VC to node J **(520)** which requests a VC to destination user U2 **(525)**. Once the VC is dedicated to node U2, U2 will acknowledge the VC to node J **(530)** which will acknowledge the VC to node H **(535)** which will acknowledge the VC to node F **(540)** which will acknowledge the VC to node C **(545)** which will acknowledge the VC to node A **(550)** which will acknowledge the connection established **(555)** to the requesting user U1. Thus a cascading set of VC requests normally occurs followed by a cascading set of acknowledgments by which a virtual circuit is set up. One should note that this requires significant processing at each of the nodes and significant network traffic in order to set up a single VC. If one is attempting to set up a thousand virtual circuits, individual handling of the virtual circuits creates a significant processing and network traffic load.

Figure 6 is a block diagram showing an operation of a network node implemented as an ATM switch. In order to quickly switch an ATM cell arriving at an incoming buffer, such as **640** to an outgoing port, a set of switching tables is maintained. Each switching table has an (incoming) virtual circuit identifier (VCI) column and an associated output port (and VCI). Virtual Circuit Identifier's are sometimes called tags. In **Figure 6**, the ports illustrated are bi-directional ports, that is, they

accommodate incoming as well as outgoing traffic. When an incoming ATM cell is received in an in-buffer for port 3, the tag (XXXXX) is read and utilized to extract the output port destination for the tagged cell using the port 3 forward table **660**. Port numbers passing to the switching interface **600** activate the switching fabric **630** to connect port 3 to port 22 during which time the contents of in-buffer **640** are passed to an out buffer **650** on port 22. In the reverse direction, the tag of an incoming cell stored in in-buffer **650** (YYYYY) is utilized to access the port 22 reverse table **670**. The input port, 3 in this case, is passed to a switching interface **600** to set up a reverse direction connection between the port 22 in-buffer and port 3 out-buffer **640**. The switch tables are set up for a very fast access and response. The contents of the switching tables a illustrated in **Figure 6** are set by the controller from the controller tables as discussed more hereinafter. In this example, the bi-directional VC takes the same route in both directions. This does not need to be the case, but is for purpose of this example.

Figures 7A, 7B and 7C show exemplary packet formats used in requesting set up of a virtual circuit bunch (VCB). The generic format for such a request is shown in **Figure 7A**. A packet header **700** begins a request with a packet type **710** indicating a VCB request. Following the

VCB request are a set of ordered fields **720A1, 720B1** and **720C1** through **730AN, 730BN** and **730CN**. Each set of the ordered set of fields specifies a destination **(720A)**, a number of VCs **(720B)** for a virtual circuit bunch or subset of virtual circuit bunch going to the destination specified in **720A**, and a level of service (LOS) **720C**. A single virtual circuit bunch request can be utilized to specify a plurality of destinations and the number of VCs associated with each destination can be arbitrarily specified.

Figure 7B shows and example of a request for a virtual circuit request from switch 1 to switch 2 for 32 virtual circuits. This would be sent as a control packet from the switch 1 controller to the switch 2 controller.

In more complex example, one which will be pursued throughout the remainder of the explanation of the invention, is shown in **Figure 7C**. Again a packet header **700** and a VCB request field **710** precede the fields specifying the virtual circuit bunch. In the example shown in **Figure 7C**, switch 1 is requesting a virtual circuit bunch totalling 25 virtual circuits, 8 of which are destined for switch 4, 8 of which are destined for switch 5, and 9 of which are destined for switch 3. When the control packet illustrated in **Figure 7C** is received at switch 2, switch 2 acts strictly ad an intermediary to traffic going to switches 3, 4, and 5. This does not

need to be the case, but is for purposes of this example. Switch 2 could easily be a destination for some of the virtual circuits. By this request, switch 1 sets up 25 bi-directional virtual circuits to switch 2. Switch 2 sets up 8 outgoing virtual circuits to switch 4, 8 to switch 5 and 9 to switch 3. In a reverse direction, the virtual circuits from switches 3, 4, and 5 are mapped to the 25 virtual circuits set up between switch 1 and switch 2. When the virtual circuit bunch is set up, no actual traffic is passed, rather the virtual circuits are defined together with their level of service and are available to pass traffic as it is routed to them from circuits originating or terminating at the end nodes of the virtual circuit bunch.

Figure 8A is a table summarizing the VCB request of **Figure 7C**. Considering columns 3 and 4 of **Figure 8A**, a virtual circuit bunch of 25 virtual circuits is set up between switch 1 and switch 2. This virtual circuit bunch is labeled VCB 3 for reference. VCB 3 is a convenient alias for referring to groups of virtual circuits, in this case consisting of 25 virtual circuits between switch 1 and switch 2. At switch 2, portions of the virtual circuit bunch VCB 3 are directed to different locations. Specifically, 8 VCs are directed to switch 4, 8 are directed to switch 5, and 9 are directed to switch 3, these subsets of VCB 3 can be labeled, conveniently,

VCB 3A, VCB 3B and VCB 3C. These labels are convenient aliases for the subset(s) of virtual circuits going respectively to destinations switch 4, switch 5, and switch 3, respectively.

Figure 8B shows the same table as **Figure 8A** except that exemplary port and VC assignments are shown.

Figure 9 is a block diagram showing the switches and links used in setting up the VCB requested in **Figure 7C** and described in conjunction with **Figure 8**. VCB 3 is established between switch 1 (900) and switch 2 (910) with 25 virtual circuits. Those 25 virtual circuits are divided up into 3 subsets with subset VCB 3A going to switch 4 (920), subset VCB 3B going to switch 5 (930), and subset VCB 3C going to switch 3 (940).

At this point in the example, the request of **Figure 7C** has been granted and the virtual circuit bunch established as illustrated in **Figure 9**. However, no traffic is flowing across the virtual circuit bunch. The virtual circuit bunches are allocated and set up in the controller tables and the switching tables of the switches but they are not being utilized.

Returning to **Figure 8B**, column 1 of **Figure 8B** shows an allocation of incoming ports and virtual circuits of switch 1 which correspond to the virtual circuit bunch VCB3 and its subsets VCB 3A and VCB 3B and VCB 3C. When a new user at switch 1 has traffic for a destination at

switch 4, switch 1 will assign the user to a VC in VCB 3A which goes to switch 4. The assigned VC in VCB 3A will immediately route the traffic to switch 4 without any setup required for virtual circuits between switch 1 and switch 4. They have been preallocated. At switch 4, the destination address of a cell is used to route the cell to its final destination.

Figure 10 is an exemplary controller table storing VCB information. This one is located at switch 1. At switch 1, VCB 3 is defined as a set of <port, VCI> tuples listed in the right hand column 1020 of the table together with the destination for those <port, VCI> tuples. The entire set of 25 VCBs may be referred to by the convenient alias VCB 3. Alternatively, subsets of VCB 3 can be referred to by the convenient aliases VCB 3A, VCB 3B, and VCB 3C, shown in the middle column 1010 each having a respective destination. Thus, a hierarchy of aliases, which may number more than the two levels shown, maybe utilized to conveniently refer to portions of a group a virtual circuits preestablished in a virtual circuit bunch.

Figure 11 shows exemplary tables in a switch controller storing connection information relating to a VCB used at a node at either end of a virtual circuit bunch. Different tables are utilized at intermediate nodes. These tables are utilized by the switch controller to establish switching table values for

controlling the switching fabric to route an incoming port and VCI to its appropriate outgoing port and VCI. As shown in **Figure 11**, different tables are utilized for the forward and reverse direction. Referring to the table shown in **Figure 11**, table **1100** which is for the forward direction differs from table **1120** for the reverse direction. Each of these table entries only shows the <port, VCI> tuples associated with the virtual circuit bunch although other entries are present. As shown in column 1130 of table 1100 has column 1 of **Figure 8B**, virtual circuits 102-106 and 111-113 from port 3 which have requested connections to destination node switch 4. Thus they are assigned a virtual circuit bunch identifier, in column 1140 of **Figure 11**, VCB 3A. The specific output <port, VCI> tuples in column 1150 to be utilized for these virtual circuits is any unused VCI on outgoing port 5 in the VCB 3A. Similarly, the other lists of input <port, VCI> tuples and virtual circuits found in column 1130 of table 1100 have an alias identifier of the virtual circuit bunch to which they have been assigned for routing data to the destination serviced by the respective virtual circuit bunch.

The reverse direction is specified in table **1120**. Since any <port, VCI> tuple associated with VCB 3 may route to any destination on switch 1 the destination address for the cell is read and the corresponding <port, VCI> specified for the destination is utilized to control

the switching fabric so that data incoming on VCB 3 is routed to the correct destination.

Considering these tables in more detail, when table **1100** is utilized and the VCI is listed as a virtual circuit bunch alias, the definition of the virtual circuit bunch is looked up in the table described in **Figure 10**, selection of an unused <port, VCI> tuple in the virtual circuit bunch is made and an unused <port, VCI> tuple is written into the forward and reverse switch tables to set up a virtual circuit connection.

Figure 12 shows exemplary tables storing connection information relating to a VCB at a node intermediate to the end nodes of a VCB. This table is one typically found in an intermediate switch, such as switch 2 shown in **Figure 9**. Table **1200** shown in **Figure 12** is very similar table **1100** shown in **Figure 11**.

Note that table **1200** shown in **Figure 12** contains many fewer entries than would be required if each virtual circuit were set up separately without the benefit of virtual circuit bunches. Table 1210 shows an even greater conciseness of expression. A single entry is sufficient to specify the reverse direction for all 25 virtual circuits in the virtual circuit bunch. In the reverse table, data incoming on any of the listed virtual circuits in column 1270 would be routed to VCB 3 in column 1260 and will be assigned in column 1250 to any unused virtual circuit in VCB 3 for transmission to

switch 1. Again to determine the unused output <port, VCI> tuple in VCB 3, one could refer to the VCB Definition table, such is shown in Figure 10 at switch 2 and then check occupancy data for the virtual circuits participating in VCB 3.

Figure 13 is flow chart of a process for setting up a VCB in accordance with the invention. The process begins when a user or node requests a VCB setup with a control packet to the local node control point **(1300)**. The local node control point parses the control packet and calculates routes to the destination(s) specified **(1310)**. This may be either a complete route calculation or a best next node calculation. The controller identifies outgoing virtual circuits needed to satisfy the contract for the level of service needed for the connections requests to each destination ~~(1320)~~. The local node control point then requests allocation of a virtual circuit bunch to the next node along with path, actually specifying the VCs desired 1320 ~~(1330)~~.

When an acknowledgment is received from the next node, an acknowledgment is sent either to the predecessor node or the requesting station 1330 ~~1340~~ and table entries are created or finalized 1340 ~~(1350)~~ thus setting up the virtual circuit.

Figure 14 is a flow chart of a process for setting up a virtual circuit using an established virtual circuit

bunch in accordance with the invention. Continuing with the previous example, the switch 1 controller receives a request for a VC to a destination switch N serviced by the VCB **(1400)**. The switch 1 controller sends a control packet to the switch N controller either over a VC of the VCB or over a control link **(1410)** requesting a connection to a destination address at switch N. The switch N controller enters controller table data for the <port, VCI> tuple to be used by the destination address, sets the switch table and acknowledges the connection by sending a control packet to the switch 1 controller **(1420)**. The switch 1 controller enters the controller table data for the destination address and the originating address and sets the switch table entries 1430 ~~(1440)~~, thus establishing an end to end connection between user 1 and user 2. Thus, in contrast to the process described in conjunction with **Figures 4 and 5**, since a virtual circuit bunch has already been set up, the only action required to establish an end to end connection is at the end points of the virtual circuit bunch. Thus, the switch 1 controller needs only establish a connection between the requesting virtual circuit for user U1 and the virtual circuit bunch destined for switch N, and switch N needs only to establish an association between the virtual circuit bunch and the destination user U2. Thus network traffic

is significantly reduced, processing at the intermediate nodes is eliminated since the virtual circuit bunch has already been set up.

Figure 15 is a flow chart of a process for breaking down a virtual circuit bunch in accordance with the invention. A virtual circuit bunch breakdown request is received on an incoming port i and routed to the controller **(1500)**. The controller consults the table for entries using aliases of the VCB being broken down and marks them for deletion **(1510)**. The connection breakdown request is forward to the next node, if any; otherwise table entries for the VCB being broken down are deleted and acknowledgment is sent to the requesting node **(1520)**. When acknowledgment of a breakdown requests is received, table entries have been actually deleted and all traces of the virtual circuit bunch are eliminated from the system **(1530)**.

Figure 17 is a block diagram illustrating sharing of virtual circuits of a virtual circuit bunch for a fast connect service. For purposes of this illustration assume that users U1A through U100A generate messages of short duration. A five circuit bunch VCB **1700** is shown going between the node servicing users U1A through U100A and the node servicing users U1B through U100B. Block **1710** shows a distribution mechanism which will assign the short duration message from users U1A through U100A to

the five virtual circuits of the VC bunch **1700** according to a distribution policy that may be partially deterministic, random, or pseudo-random. In any case, at the destination end, **1720**, the destination addresses of the cells are read and the cells routed to the destination address without a formal virtual circuit being setup. In short, when one of users U1A through U100A has a message to send, it is assigned to one of the virtual circuits of virtual circuit bunch according to the distribution policy without actually setting up a connection and it is routed at the receiving end to the destination, also without setting up a connection. Thus, the five virtual circuits of the virtual circuit bunch can service a much larger number of users without a separate on the fly connection establishment. This provides a very quick and efficient fast connect service.

Figure 18 is a flow chart of the process described in conjunction with **Figure 17**. When a fast connect packet plus request is received at a node, a <port, VCI> tuple from a VCB going to the destination node is selected ~~(1700)~~ 1800. The fast connect packet/request is sent over the selected <port, VCI> tuple to the destination node ~~(1710)~~ 1810. The fast connect packet/request is routed at the destination node to the specified address ~~(1720)~~ 1820 and an optional fast connect acknowledgment is returned

to the originating address in the same way fast connect packet/request was sent (1730) 1830.

Figure 19 is a block diagram of a portion of a network used to illustrate split routing of a VCB. Assume that node A requests 150 virtual circuits to destination E at a given level of service (LOS). Only 100 such virtual circuits are available through C and node D has only 85 available. This decision may also be the result of a policy decision (for example, load-balancing) of the network control system. Node B can allocate a virtual circuit bunch routing a portion of the VCB over a node C and the remainder over node D. This kind of split routing was not a capability provided by the virtual path technology of the prior art. Thus VCBs can split the routing of a virtual circuit bunch into convenient units.

The same figure illustrates another benefit achieved by the invention. In the prior art, there is sometimes a need to send keep alive packets or refresh packets periodically over each virtual circuit to avoid a connection timing out and being broken down. With a virtual circuit bunch, one only needs to send one keep alive packet or refresh packet per virtual circuit bunch and not one for each circuit. This reduces network traffic and processing power requirement.

Figure 20 is a block diagram showing the cell interleaving problem and how it is solved when using a virtual circuit bunch. The problem arises when, as shown in **Figure 20**, two <port, VCI> tuples attempt to send to the same destination. Thus, the port 1 switch table 2010 attempts to route VCI 120 to port 2 VCI 150. Similarly, as shown in table **2020**, port 4 attempts to route VCI 125 to outgoing port 2 and VCI 150. Thus, cells from two sources would occur on the same outgoing port and VCI and would result in a error condition at the destination. Such a failure will be detected when the CRC result will not conform to the expected result. If outgoing ports 2 and 6 were part of the same virtual circuit bunch, one of the two conflicting connections could be routed by the switch fabric 2060 over another unused VCI of the virtual circuit bunch or over an unused VCI of port 2. Alternatively, some of the virtual circuits could be used to permit sharing by appropriate selection as discussed above. Thus, when cell interleaving occurs, rerouting can be easily accomplished using a virtual circuit bunch and cells successfully routed to the correct destination without error.

The previous examples illustrated VC bunching is somewhat static manner - the bunches were established as a unit, and they remained the same through the life-time. However, VC bunches can also be established and modified

dynamically. The system may decide that a set of already established VCs (and bunches) can be served more efficiently by aggregating them together into a large bunch. Alternately, when a new VC (or bunch) establishment request arrives, the switch may decide to incorporate it into an existing bunch (by expanding it). Such dynamic aggregation may require changes in VC identifiers.

Figure 21A illustrates this concept; initially, the switches A, C, D, E and F carry a bunch of size 32. Now, node B wants to set up a bunch of size 32 to node G. When the new bunch request arrives at node C (from node B), node C aligns the new VC bunch with the existing bunch, to create a size 64 bunch at entry to node D as shown in **Figure 21B** which sets up the expanded bunch of 64 to node E, where the two bunches are separated once again (as the paths diverge). In this manner, one can dynamically create flexible bunches via aggregation.

The invention disclosed provides improved efficiency for setting up multiple connections in a network environment. It results in a reduction in the virtual circuit table size. It maintains compatibility with current switches without requiring hardware changes to participate in the VC bunching. It permits use in multi-path virtual circuit bunches which was not feasible in the case of virtual paths. It supports aggregation by

which data paths may start out as individual circuits but later the neighbors may agree to change identifiers so as utilize VC bunching. It provides efficient support for sharing resource allocations. It supports a variety of techniques for handling ATM cell interleaving problems, particularly in multi-sender multi-cast. It provides efficient support for fast connection set up and for a fast connect service. It provides considerable efficiency for handling keep-alive and refresh packets. It does not require a full 2^{16} address space required by virtual paths. Only the address space required for a particular virtual circuit bunch is set up. It allows virtual circuit bunches to be subdivided to service multiple destinations. Once a virtual circuit bunch is set up essentially zero latency is required for set up of a virtual circuit from the origin to the destination. With virtual circuit paths, once the large address space was allocated, resources were assigned and dedicated to it. Those resources were not used, the resources were wasted. Virtual circuit bunches permit allocation of resources on an as needed basis.

Thus, the techniques of the invention described overcome the problems of the prior art and permit great flexibility in the establishment and use of virtual circuits.

Although the present invention has been described and illustrated in detail, it is clearly understood that the same is by way of illustration and example only and is not to be taken by way of limitation, the spirit and scope of the present invention being limited only by the terms of the appended claims.